



## DOWN UNDER CLUB

P

Editor  
Harry Huggins  
12 Thomas Str.  
Mitcham. 3132  
03-873-1408

Treasurer  
Ron Allen  
2 Orlando Str.  
Hampton. 3188  
03-598-4534

P

A MERRY CHRISTMAS TO ALL OUR MEMBERS.

In the mailbag this week was a pair of 'P' plates, sent by the Editor of our 'Competitor'.

Now why 'P' plates. Have they taken them off their 'PREMIUM' state, or is it a hint that VZDU should drive on 'P' plates? Cheeky So & SOs

The intention was neither. In fact they make a very good packing when sending disks through the post. I hasten to add though, that if you use them, make sure they are the old ones and NOT the newer magnetic type, or you may be sending a blank disk.

But I am displaying them as PROVISIONAL.

Fact is that I have had a stroke and am looking for a new Editor. (See the Trading Post for details of this lucid position. A touch of insanity would also be a help in having your application accepted.) After 4 years I think that someone else should take up the torch. I suggest someone younger. Much younger. If someone cares to take over I will give them all help for the first few issues, and all the data we hold and programs to carry on with. They will also have first option to purchase all or any of my equipment. There are sufficient funds to carry on, which we will transfer to them, unless Ron cares to carry on as treasurer. That is not for me to say.

I shall put out one more issue at least. Jan/Feb. After that depends on circumstances. So the "Ball is in your corner". There is too much "Let George (read Harry) do it" Well the old grey mare ain't what she used to be. I have enjoyed doing this, but of late support has fallen off to zero, and I don't even know if the rag gets read!

Failing someone taking over I have made a suggestion to HVZUG that we combine the 2 clubs, or the 2 N/Letters. I shall be in Newcastle over Xmas and will discuss it with Joe. To put your minds at rest, if we do close up, there are sufficient funds to refund everyone their outstanding credits, So you won't be ripped off, as has happened in times past.

Let us know what you think and any or all suggestions you may have. I will put out one more issue. It is your club so now take over and do something about it. I am sorry. But I can't go on pulling articles out of either my hat or from Bob. Even he must eventually run out.

Now turn to the trading post and write your application.

FOR SALE  
\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$

VZ300 w/16K mem.expan. DISKDERIVE and CONTROLLER  
PRINTER INTERFACE. ALL PLUG PACKS AND LEADS. \$250.

-----  
VZ300 WITH EXPANDED ROM CHIP FITTED  
DISK DRIVE  
DISK DR. CONTROLLER WITH EXPANDED DOS CHIP FITTED  
PRINTER INTERFACE AND ALL LEADS AND POWER SUPPLIES.  
\$350.

-----  
VZ300 COMPUTER----\$50.

-----  
MEMORY 64K-----\$75

-----  
MEMORY 200 16K--\$25

-----  
MEMORY 300 16K--\$35

-----  
PRINTER INTERFACE NEW ---\$30. USED \$20

-----  
WORDPROCESSOR CARTRIDGE.-----\$50

-----  
DISK CONTROLLER FITTED WITH EXT. DOS CHIP AND RESET.\$50

-----  
DISK DRIVE-----\$70.

-----  
DISK DRIVE POWER SUPPLY.----\$40.

-----  
VZ300. NOT WORKING. 4only. \$20 ea.

-----  
LIGHT PEN----\$50

-----  
JOYSTICKS---\$35 PR.

-----  
PLUG PACK ~~9V~~<sup>12V</sup>---\$15.

-----  
CASSETTE RECORDER DR20----\$25  
" " DES 7344- new---\$30.

-----  
DISK STORAGE BOXES---\$10.

-----  
BACK ISSUES VZDU. \$1.50 EA POST FREE

-----  
VZ ORIGINAL GAMES TAPES. \$3 ea.

-----  
PRINTER SEIKOSHA GP250X. Tractor feed.---\$75.  
-----



# GAMES COLUMN

Welcome to the Christmas issue of the games column.

I've just finished my yr 11 exams and already I have assessments to complete before week something-or-other. Well, lacking any sort of energy and will power I've let the games hints and tips slip from my mind. But I did have a games review prepared for a few issues ago which did not make it to the final drawing board.

So for this issue I will review a game that I'm sure everyone has heard of; "DAWN PATROL."

It is in binary and I can not enter into the program to identify the author. It is very popular, as anyone who has followed the high scores from the beginning would know.

You are given the task of rescuing 80 prisoners from four prison camps behind enemy lines. Using the only available means of transport, you fly a helicopter to each camp, land inside the enclosure and wait till all the prisoners, or as many as you can carry, run out to the chopter. But these's a few catches, the more prisoners you collect the more petrol you use up. Camps one to four are guarded be tanks. Tanks can only shoot parrallel to the ground and can get you while you are loading the prisoners. Camps one to three are also guarded by missiles launches, which shoot missiles into the air and travel at 4 degrees left or right. Camps one and two also have 'planes that appear, fire a missile and disappear into the distance. You can not shoot the 'plane but you can shoot the missile. A few times more then a normal surface to air missile.

On top of that your mission starts at 4.00am and terminates at 6.00 am.

You score points for the amount of prisoners you return to base with. Prisoners are worth more depending on which camps they came from. Camp four (the closest) prisoners are worth 100, camp three are worth 200, camp two are worth 500 and camp one are worth 1500.

O.K so it's a pretty short review. How about some-one, ANYONE, giving me some fill-in material. You know ... an opportunity to get your name up in "lights." A drawing or design would do, using one of the drawing programs from our library. It could be a games title, or your version of a little gobbling games monster, or whatever catches your imagination. But it would be a help for fill-in material!

Library tape two has a drawing program called "Sketches". It will print out your drawing. The others won't print, but it is easy to modify them so that they will. Load the program you want, <LIST> it, and find the string commands, then add the print or copy command. Just take lines 3020-3050 in "Sketches" as a guide. These lines contain the string commands. Line 3030 contains the string which will enable you to print your design. If "P" has already been used for something else then just use another letter. Make a note WHICH letter, because if you hit the wrong one you will lose your drawing

and have to start ALL OVER AGAIN.

Don't forget to load "Printer Patch" before you load the drawing program.

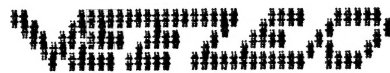
Sorry to hear you haven't been well, Harry. But it's the timing of your misfortune that has me worried as well. Just a week beforehand my younger brother Mitch played soccer at South Melbourne. He was delighted to be able to meet Harry at the soccer grounds. THEN, one week later ... oh, dear! As an older brother I have always known of the CURSE of younger brothers!!!

Just kidding everybody! HONEST!!!

Anyway Harry, here's hoping you're feeling great again now and that you and all VZ-ers have a tremendous Christmas and a safe and happy New Year.

TIM.

## TRADING POST.



DOWN UNDER CLUB  
12 THOMAS ST. MITCHAM. 3132.

### SITUATION SOON TO BECOME VACANT.

Editor of Leading V.Z. Newsletter and Secretary of old established Computer Club.

Any 12 year old person, with Master's Degree in Electronics, Computer Languages, Psychology and Psychiatry, and with the patience of Job, the foresight of Jacob, the manipulating ability of Mandrake and the luck of James Bond would be acceptable.

However, greater Age and lesser qualifications would be considered.

Location; Anywhere

Hours; Enormous

Salary; Nil

Any applications to President, care of this Rag.

# INTRODUCTION TO PROGRAMMING — PART I

by Bob Kitch

I have been asked to contribute a series on BASIC programming for the VZ-Users. So here goes.

Firstly, the series will be unconventional. Most introductions to BASIC proceed blow-by-blow through the various BASIC commands. I will not - many texts exist which can explain these better than I can.

Secondly, the series will initially be non-specific to any particular computer language. General programming concepts and guidelines will be offered. The principles will be equally applicable to BASIC, Assembler, Pascal or whatever.

Thirdly, advanced programming concepts and hints will be offered as they are needed. This is the best time to introduce these since their mystique is removed.

Fourthly, early emphasis will be on PLANNING, ORGANISING and MAINTAINING a program, rather than encouraging feverish coding at the keyboard (which is usually commenced too early by beginners).

It is quite possible to recognise a breed of compulsive programmers, born from the home micro boom. This breed, is emerging from the brave new world of tomorrow's technology whose reason for existence is simply to program. People become totally fascinated by the unlimited abstract world that the inside of a computer offers. We can create a Universe or any World inside a machine. In the abstract world of programming, a well thought out programming method serves as a MAP, and the techniques of software engineering are the WEAPONS. These then are the main threads of this series.

Let's commence this month with a few definitions and concepts to ponder over until the next installment.

THE COMPUTER is a machine, and is only capable of doing simple work. It has been termed by some as "a remarkably efficient counting machine with a large memory - but no brains!" It has no intelligence and cannot think.

A COMPUTER SYSTEM consists of four elements :-

1. the Central Processor Unit (in the VZ it is the Z-80A microprocessor chip) with "primary memory" (ROM and up to 34K RAM).
2. Input devices - keyboard, cassette, disk and so on.
3. Output devices - screen, printer, cassette, disk, in-built speaker, voice and sound synthesisers etc.
4. "Secondary memory" - not essential but may be cassette or disk when used to update or relieve primary memory.

MAN-MACHINE INTERFACE. The interaction between man-machine inputs and outputs is a continuous and circular feedback process. e.g. man output (keypress) is machine input ..or.. machine output (screen prompt) is man input- ..and so on. This interaction forms the basis of using computers.

The four fold subdivision of a computer system is little different to our own mental capabilities. The CPU and primary memory is broadly equivalent to our mind. The I/O devices are similar to our senses (touch, taste, sight, sense of heat, speaking, hearing). The secondary memory is directly comparable to our use of external aids to assist our memory, such as note books, filing cabinets of information, telephone directories - all of which have slow access and are difficult to recall compared with things already resident in our mind.

COMPUTER PROCESSES or CAPABILITIES are surprisingly few in number. There are only FOUR and unless an exercise or problem can be broken down into these elementary processes, then coding of the program should not commence. A greater understanding of the problem is required before proceeding.

It is important to clearly distinguish two things whilst programming. The first, is to devise a LOGICAL solution to the programming exercise, which is

quite independent of the particular language to be used. The second, is the actual CODING of the exercise being undertaken. The latter stage is easy, provided that the former is well understood. The computer program will only function correctly if the logic of the program is correct, and there are no aids or diagnostics available from the machine to assist in achieving correctness in this demanding aspect of program design. Some diagnostics are however available to assist in the coding portion of the task - such as the SYNTAX checking.

As one becomes more familiar with programming languages it is soon apparent that many of the powerful command structures are simply macro instructions formed from these few "primitives".

The four processes are :-

1. Input data and store it in primary memory - the data may be either "raw" data input (e.g. from keyboard) or read-in from the secondary store. (e.g.tape).
2. Output data already stored in primary memory - either as "output" (e.g. to screen) or written-out to secondary memory (e.g.tape).
3. Perform simple arithmetic procedures (addition or subtraction) upon data in primary memory only.
4. Perform logical comparisons (disjunction, conjunction and negation) between two items of data in primary memory.

(Remember - I/O, arithmetic, comparisons only)

To continue the analogy with ourselves, I doubt whether we can do anything more than these operations except that we use experience. The computers' analogue of this is the PROGRAM as it possesses zero intelligence.

THE PROGRAMMING TASK is to utilize the high speed and large memory capacity of a computer system to do something useful - such as carry out calculations (number crunching), play games, monitor house security etc.

The spectrum of tasks involved in programming is very broad, so little wonder that beginners have trouble grasping the essentials, or that many programs are "badly" written. The task involves taking an idea or concept and translating that into a symbolic (program statement) form of representation. An intermediate stage in this translation often involves modelling the phenomenon being programmed. This psychologically involves moving from concrete concepts to various levels of abstraction - again a very difficult thing for, particularly young, minds to master.

The transition from an idea to a program can seldom be achieved in one leap - more often a number of intermediate steps are required. Liken it to writing an essay where drafts and notes are used before the final prose is produced. Fortunately a number of useful tools have been developed to assist in producing a good program.

In my view, one of the greatest pitfalls of the home computer boom is that these intermediate steps are not understood by Users so that, at least, bad programs and, at worst, disillusioned programmers result. Many of these people may find their way into the computer industry of the future. There is ALWAYS more personal satisfaction in achieving a "good" job even if it is only a games program for the kids. It is also more fun, (the essence of home micros) as there is less hassle in getting a program to run, and more time for more programs.

In the microcomputer environment where there are always hardware limitations, it means that it is very difficult to completely separate hardware and software aspects of the programming task. The programmer may have to get "close to the hardware" - usually due to hardware/memory limitations or restricted I/O capabilities. Don't shy away from hardware by saying "but I am only interested in writing programs" as the two are somewhat inseparable.

Next month we will look at the various stages in the programming task, or how to approach a programming exercise. (see, no mention of BASIC code in this article!)

Finally, I would like to offer to Users that your programming queries will be

answered if you write to me - with a SAE, please. In this manner you should get what you want and I will obtain a feel for the type of problems Users in the Hunter Valley are experiencing.

Write to Bob Kitch, 7 Eurella St., KENMORE, Qld. 4069

As mentioned in Part 1 of this series, the programming task is a large and complex feat of organization and requires a wide range of skills. It is possible, and best, to break the task down into six segments - each of which must be thought about, planned and then carried out to ensure the successful completion of a software project. Even a small program requires that a cursory consideration of the six segments be made - although some of them may be quickly passed over as trivial. But it is certain that larger programs (more than 200 lines) require careful planning for success.

Before describing the six steps, it is worth thinking about "What makes a GOOD program?"

A program may be judged from a number of different standpoints; each is not necessarily mutually exclusive and sometimes some conflicts require that a trade-off be made.

The first criteria is that a program should be EFFICIENT. Efficiency can be considered from a number of varying view points. For example, optimization of the run-time can be considered as efficient. Also, reduction in storage requirements for both program code and variables can be considered as efficient programming. Furthermore, and particularly if one is developing software commercially, then efficiency can be measured in terms of the actual time required to get an applications program running and the ease of maintenance of that code. The use of appropriate data types and data structures can greatly improve the efficiency of a program. The selection of a suitable algorithm can also assist. Finally, ease of debugging so that the program can be updated or modified may be considered desirable.

The second criteria is GENERALITY and it is here perhaps that so many programs "score" so poorly. Rather than a program being written to solve a particular chore, it should be broadly written to handle a wide range of problems. The use of subroutines and functions developed and debugged previously can enormously improve programming productivity. Often a simple substitution of a variable for a constant in a program can broaden the the applicability of the program significantly.

The final criteria is ELEGANCE, which is a little harder to both define and achieve. An elegant program is one that is simple and ingenious, and possibly uses an algorithm or data structure that may not be immediately obvious to the application. The so-called "programmer's tricks" are often elegant solutions to a programming problem; but beware, some are attempts by programmers to conceal their programming strategy.

These then, are general guidelines to try and attain in your programming and by which to judge a particular programming effort as good, mediocre or poor. Notice that they are not language specific comments and are equally applicable to any programming language or exercise.

To return to the six steps in the programming task - I will briefly discuss each in turn and ask that you consider each one when embarking upon your next programming exercise. Also as one proceeds through the steps, it is often necessary to recycle back through some of the preceding steps, to iteratively improve the exercise and your understanding of ideas.

1. PROJECT SELECTION. This may appear trivial, but we all have too many ideas for programs and rarely know which one to tackle next. Also be honest with yourself; some of the projects are probably too ambitious for your existing skills and an attempt upon these will possibly result in frustration and perhaps failure. Choose an exercise that is challenging and worthwhile. Try not to "reinvent the wheel", try to be aware through reading magazines or discussing with other Users what programs are already available. Modifying an existing program to suit your specifications is sometimes quicker - it also allows you to study how other programmers tackle problems. O.K., so now you have an idea or problem that you wish to tackle and solve.



2. PROJECT FEASIBILITY. Again be honest. Do you have the hardware, software and know-how to achieve the result? It is not really much use trying to write large business-oriented data base programs for an 8K taped-based VZ! Check that the task is reasonable.

3. PROJECT DEFINITION. This is where the idea starts to get translated into a reality. It is also the phase where generality can be written in. It is easiest to start by thinking about the input to the program. Is it keyboard oriented, or is it to come from a programmable I/O port? Perhaps the program reads only DATA statements to configure itself or maybe the program must check if a printer is connected to the system? Start defining what the input will look like. Assign variable names with meaningful mnemonic names at this stage also.

Next, define the output expected from the program. Is it to write to tape and in what format? Perhaps it is to be screen oriented - can sound be used - or perhaps voice synthesis to tell the operator what is going on? Plan very carefully and fully the layout of the expected output as this is how Users will initially perceive the quality of the program.

After defining the I/O for the program we should now have a feel for the anticipated range of parameters that the program is meant to accept and also handle. This brings in the very important concept of defining the BOUNDS within which the program must function correctly. Following on from this, is range checking of all input parameters so that the program cannot go beyond the range that it was designed for and give unexpected results. A number of warning messages must be built into the program along with error capture and recovery routines. It is failure to define the operating bounds of a program that causes most crashes or rogue behaviour. Even the definition of integer variables at this stage can assist by improving program execution time and reducing storage requirements.

The definition stage should be roughed out on pieces of paper kept for later reference. Perhaps better, is to use an old exercise book. Another benefit of this is that over a period of months your progress can be measured and your growth of programming ideas recorded. Another benefit (although I hardly dare mention it!) is that if, after the coding stage, a system crash occurs and you didn't SAVE the program, then all is not lost - at least an outline of the program remains.

4. DESIGN PHASE. Having sorted out I/O and operating bounds, the actual selection of an algorithm to achieve the result is commenced. By this time some idea of the number of variables required and their type should have begun to gel. This is also the stage where your basic honesty in stages 1 and 2 may catch up with you! Data structure organisation and algorithm selection are really experience-related skills - hence the suggestion to read and/or modify existing programs. But do not despair - practice makes perfect.

5. IMPLEMENTATION PHASE. To date very little actual coding should have been done; in fact the computer need not even have been turned on! Some people may be surprised at how late in the task the computer actually enters into the picture. An awful lot of planning and organizing can be done off the computer and on the "backs of old envelopes".

It is also at this stage that the choice of programming language should be made. Is the program time dependant? If it is, then it should probably be written in Assembler. If the actual timing is not so critical then writing in BASIC with its diagnostics and helpful features (so typical of a high level language) deem it sensible. Experienced programmers will probably use a bit of each in practice. A very sensible compromise is to develop the program in interpreted BASIC and once finalized and debugged, compile the BASIC code to speed up execution.

6. EVALUATION PHASE. This is the moment of truth! Does the program fulfil all the criteria set out in the definition phase. If so, then you have successfully achieved your task. Is the output as you expected it? Are the results correct? It is a good idea to have a standard set of data to exercise the program so that it can be quickly verified after a program alteration. Ensure that all logical paths through the program have been exercised so that no spurious errors of logic



remain undetected. Finally, deliberately try values that are out of the intended bounds of the program to ensure that you have trapped them and that the program recovers from this type of misuse above and beyond its' intended design range.

CONTINUED FROM PAGE 14

0561*VZ200 GRAND PRIX	*	0590	DEFW 8686H
0562MSG2 EQU \$		0591	DEFW 8686H
0563*BY P. HICKMAN	*	0592	DEFW 8686H
0564MSG3 EQU \$		0593	DEFW 8686H
0565*** FUNCTION KEYS ** *		0594	DEFW 8686H
0566MSG4 EQU \$		0595	DEFW 8686H
0567* Q = INCREASE SPEED		0596	DEFW 8686H
0568MSG5 EQU \$		0597	DEFW 8686H
0569* A = DECREASE SPEED		0598	DEFW 8686H
0570MSG6 EQU \$		0599	DEFW 8686H
0571* M = MOVE LEFT	*	0600	DEFW 8686H
0572MSG7 EQU \$		0601MS14 EQU \$	
0573* , = MOVE RIGHT	*	0602*YOUR SCORE :=*	
0574MSG8 EQU \$		0603MS15 EQU \$	
0575*PRESS B TO ENTER BASIC		0604*PRESS ANY KEY TO CONTINUE	
0576MSG9 EQU \$		0605MS16 EQU \$	
0577*PRESE ANY KEY TO START		0606*YOU ARE THE CHAMPION	
0578MS10 EQU \$		0607;#####	
0579*CHAMPION SCORE =	*	0608;IMPROVEMENTS NEEDED	
0580MS11 EQU \$		0609;#####	
0581*2ND BEST SCORE =	*	0610;ADD JOYSTICK CONTROL	
0582MS12 EQU \$		0611;ADD BREAK TO RESTART	
0583*3RD BEST SCORE =	*	0612;ADD FIRE BUTTON/RESTART	
0584MS13 DEFW 8686H		0613;ADD MAX NO OF CARS = 5	
0585;VERGE DESIGN		0614;ADD 5 LEVELS OF PLAY	
0586	DEFW 8686H	0615;HIGHEST LEVEL HAS MORE	
0587	DEFW 8686H	0616;OBSTACLES, FASTER SPEED,	
0588	DEFW 8686H	0617;AND NARROWER ROAD	
0589	DEFW 8686H	0618;ADD TITLE SCREEN	
		0619;DISABLE BLOAD COMMAND	

## HISTORY

Reprinted from the MPCUG journal PC UPDATE.  
Author Ian McDowell

Australia possessed only three electronic digital computers in the late 1950s;

.CSIRAC at Melbourne University

.SILLAC at Sydney University

.WREDAC at the Weapons Research Establishment at Salisbury

CSIRAC is now a museum piece, and you may see it at Monash University Christholme Campus in Caulfield.

CSIRAC used punched paper input. It possessed only addition, subtraction and multiplication resident routines. All other required copying library tape segments to the user's input roll. Logical circuits used thermionic valves, the transistor had appeared but not been mobilised. Mercury delay line held information. Cathode ray tubes displayed the bytes contained in fifteen storage registers. A Friden Flexowriter read and printed punched paper tape output. CSIRAC had about the power of a T159 TEXAS Instruments programmable calculator. Nonetheless, on the word of command DO, it produced RESULTS. It freed us from slow and noisy electro-mechanical machines. Learned persons produced all sorts of worth-while discoveries on all three computers. The first conference on automatic computing and data processing in Australia at Sydney University in May 1960 gave opportunity to share these discoveries. Presenters gave a total of 157 papers; 42 on commercial applications, 65 on technical applications, 41 on design and programming techniques and 9 describing other computers soon to be offered by commercial firms. This writer spoke to a technical paper on a problem solved using CSIRAC which various learned journals later published in Australia and overseas. The rise of electronic digital programmable computing seems rapid to its youthful enthusiasts, but it has taken a third of a century to reach its present level, and its early proponents look for the 486 in the geriatric ward.

Postscript by Peter Smith, editor of MPCUG PC UPDATE.

I too have fond memories of CSIRAC--my introduction to computing. I recall those huge festoons of 12-track paper tape and the desire, and need, to cram more and tighter code into its 768 words of memory (20 bits each, yes the "BYTE"--a word not yet invented--was then only 5 bits long). I compare my portable, battery powered telephone, weighing a few ounces, to the tons and cubic yards of CSIRAC, needing a small power station to drive it, and realise that the phone has more memory than CSIRAC and is much more user friendly! I have long since decided that only knaves and fools dare predict longer than about 5 years in this industry.

# GRAND PRIX

```

0001;VE FILE VERSION 1
0002;WRITTEN IN 1987
0003;BY PETER J. HICKMAN
0004;FROM MICHIGAN MAGAZINE
0005;VOL 3 NUMBER 7 (JUNE 1982)
0006;#####
0007;A THE-80 PROGRAM
0008;#####
0009;SET YOUR OWN START ADDRESS
0010;SO THAT IT IS THE SAME AS
0011;THE ORIGIN FOR YOUR PROGRAM
0012SADR EQU 0C000H
0013;SET ORIGIN TO 0C000H
0014;SET PARAMETER N=3
0015;#####
0016;SELECT MODE;(0)
0017ENTR XOR A
0018 LD (7896H),A
0019;NORMAL PRINT
0020 LD (7818H),A
0021;NORMAL BACKGROUND COLOUR
0022 LD (7819H),A
0023;OUTPUT DEVICE = VIDEO
0024 LD (789CH),A
0025 INC A
0026;NORMAL INPUT
0027 LD (7896H),A
0028;USR ADDRESS
0029 LD HL,SADR
0030 LD (789EH),HL
0031;T.O.N. POINTER
0032 DEC HL
0033 LD (78B1H),HL
0034;CLEAR 50 BYTES
0035 LD DE,OFFCH
0036 ADD HL,DE
0037 LD (78A0H),HL
0038;START GAME : PAUSE
0039STRT LD BC,2FFFH
0040 CALL 0060H
0041;CAR START POSITION
0042 LD DE,VIDE+488
0043 LD (CARL),DE
0044;ROAD START POSITION
0045 LD DE,VIDE+7
0046 LD (POSN),DE
0047;INITIAL SPEED
0048 LD DE,1800H
0049 LD (SPED),DE
0050;INITIAL SCORE
0051 LD DE,0000
0052 LD (YSCR),DE
0053;CLEAR SCREEN
0054 CALL 01C0H
0055;WRITE INTRODUCTION
0056 CALL FRAM
0057;SORT SCORES
0058 CALL SORT
0059;WRITE SCORES TO SCREEN
0060 (CALL, F---)
0061;WAIT FOR (BOARD INPUT
0062 CALL AYB
0063;ENTER B...IC IF 'B' PRESSED
0064 CP 88
0065 JP NZ,CLES
0066 JP 0069H
0067;
0068;

```

```

0070;
0071;CLEAR SCREEN
0072CLRS CALL 01C0H
0073;DRAW VERGE & ROAD ON LINE 1
0074 CALL DRAW
0075;COPY LN 1 TO ALL SCREEN
0076 CALL COPY
0077;DRAW CAR
0078 CALL DRCR
0079;SCROLL ROUTINE
0080;SCROLL DOWN BY ONE LINE
0081SCRL LD HL,VIDE+479
0082 LD DE,VIDE+511
0083 LD BC,479
0084 HALT
0085 HALT
0086 LDDR
0087;RANDOMLY BEND ROAD TO
0088;LEFT OR RIGHT
0089 LD HL,2
0090 CALL RAND
0091 LD A,E
0092 LD DR,(POSN)
0093 CP 01
0094 JR Z,LFT1
0095 INC DE
0096 JR PRNT
0097LFT1 DEC DE
0098;KEEP ROAD ON SCREEN
0099PRNT LD HL,VIDE+15
0100;COMPARE HL WITH DE
0101;A=0 IF HL=DE
0102;A=1 IF HL>DE
0103;A=FF IF HL<DE
0104 CALL 0A39H
0105 CP 01
0106 JR Z,LEST
0107 DEC DE
0108LEST LD HL,VIDE+1
0109 CALL 0A39H
0110 CP OFFH
0111 JR Z,OKOK
0112 INC DE
0113;SAVE ROAD POSITION
0114OKOK LD (POSN),DE
0115;DRAW VERGE & ROAD TO LINE 1
0116 CALL DRAW
0117;1 IN 10 CHANCE OF
0118;OBSTACLE ON ROAD
0119 LD HL,10
0120 CALL RAND
0121 LD A,E
0122 CP 01
0123 CALL Z,OBST
0124;INCREASE SPEED
0125 LD DE,(SPED)
0126 DEC DE
0127 DEC DE
0128 LD (SPED),DE
0129;INCREASE SCORE
0130 LD HL,(YSCR)
0131 INC HL
0132 LD (YSCR),HL
0133;KEYBOARD SCANNING ROUTINE
0134;SCAN FOR ',' KEY
0135SCAN LD A,(88EFH)

```

//



```

0281 CALL 14C9H
0282; CONVERT RESULT TO INTEGER
0283; DE = RND(HL)
0284 CALL 2B05H
0285 RET
0286; MOVE RIGHT
0287MVR1 LD DE, (CARL)
0288 INC DE
0289 LD (CARL), DE
0290 RET
0291; MOVE LEFT
0292MVLE LD DE, (CARL)
0293 DEC DE
0294 LD (CARL), DE
0295 RET
0296; INCREASE SPEED
0297INSP LD HL, (SPED)
0298 OR A
0298 LD DE, 50
0300 SBC HL, DE
0301 LD (SPED), HL
0302 RET
0303; DECREASE SPEED
0304DESP LD HL, (SPED)
0305 LD DE, 50
0306 ADD HL, DE
0307 LD (SPED), HL
0308 RET
0309; SORT SCORES
0310; COMPARE YOUR/CHAMPION SCR
0311SORT LD HL, (YSCR)
0312 LD DE, (CSCR)
0313 CALL 0A39H
0314 CP OFFH
0315 JR Z, NXT3
0316; YOUR SCORE IS CHAMPION
0317 LD (CSCR), HL
0318 KK DE, HL
0319; SET CHAMPION SCORE FLAG
0320 LD A, 1
0321 LD (MSG), A
0322; COMPARE YOUR/SECOND SCORE
0323NXT3 LD DE, (SSCR)
0324 CALL 0A39H
0325 CP OFFH
0326 JR Z, NXT4
0327; YOUR SCORE IS SECOND SCORE
0328 LD (SSCR), HL
0329 EX DE, HL
0330; COMPARE YOUR/THIRD SCORE
0331NXT4 LD DE, (TSCR)
0332 CALL 0A39H
0333 CP OFFH
0334 JR Z, RSLT
0335; YOUR SCORE IS THIRD SCORE
0336 LD (TSCR), HL
0337RSLT LD A, (MSG)
0338 CP 1
0339 JR NZ, MSG
0340; PRINT CHAMPION SCORE
0341 LD HL, MS16
0342 LD DE, VIDE+388
0343 LD BC, 20
0344 LDIR
0345; YOUR SCORE NOT CHAMPION
0346MSG LD HL, MS15

```

```

0350;
0351 LD DE, VIDE+452
0352 LD BC, 25
0353 LDIR
0354 RET
0355; SOUND ROUTINE
0356SND PUSH AF
0357 PUSH BC
0358 PUSH DE
0359 PUSH HL
0360 LD HL, 0064H
0361 LD BC, 000AH
0362 CALL 345CH
0363 POP HL
0364 POP DE
0365 POP BC
0366 POP AF
0367 RET
0368; DRAW OTHER CARS ON ROAD
0369OBST LD HL, 11
0370 CALL RAND
0371 LD HL, (POBN)
0372 ADD HL, DE
0373 KK DE, HL
0374 LD HL, CAB1
0375 LD BC, 2
0376 LDIR
0377 RET
0378; WAIT FOR KEYBOARD INPUT
0379; RESULT WILL BE IN 'A' REG
0380KEYB PUSH BC
0381 PUSH DE
0382 PUSH HL
0383 CALL 0049H
0384 POP HL
0385 POP DE
0386 POP BC
0387 RET
0388; PRINT INSTRUCTIONS
0389FRAM LD HL, MSG1
0390 LD DE, VIDE+4
0391 LD BC, 20
0392 LDIR
0393 LD HL, MSG2
0394 LD DE, VIDE+36
0395 LD BC, 20
0396 LDIR
0397 LD HL, MSG3
0398 LD DE, VIDE+68
0399 LD BC, 20
0400 LDIR
0401 LD HL, MSG4
0402 LD DE, VIDE+100
0403 LD BC, 20
0404 LDIR
0405 LD HL, MSG5
0406 LD DE, VIDE+132
0407 LD BC, 20
0408 LDIR
0409 LD HL, MSG6
0410 LD DE, VIDE+164
0411 LD BC, 20
0412 LDIR
0413 LD HL, MSG7
0414 LD DE, VIDE+196
0415 LD BC, 20
0416 LDIR

```

```

0140;
0141 JR NZ, LEFT
0142 CALL MVRI
0143 JR SPCT
0144; SCAN FOR 'M' KEY
0145 LEFT CP ODFH
0146 JR NZ, SPCT
0147 CALL MVLE
0148; SCAN FOR 'Q' KEY
0149 SPCT LD A, (88FEH)
0150 CP OEFH
0151 JR NZ, BRKE
0152 CALL INSP
0153 JR NMOV
0154; SCAN FOR 'A' KEY
0155 BRKE LD A, (88FDH)
0156 CP OEFH
0157 JR NZ, NMOV
0158 CALL DESP
0159; GET LOCATION OF AREA
0160; IN FRONT OF CAR
0161 NMOV LD HL, (CARL)
0162 LD DE, 32
0163 SBC HL, DE
0164; CHECK IF ROAD IS CLEAR
0165 CHK1 LD A, (HL)
0166 CP OBFH
0167 JR Z, CHK2
0168 CP OB7H
0169 JR Z, CHK2
0170; SET CAR CRASHED FLAG
0171 LD A, 01
0172 LD (FLAG), A
0173 JR DCAR
0174 CHK2 INC HL
0175 LD A, (HL)
0176 CP OBFH
0177 JR Z, DCAR
0178 CP OB7H
0179 JR Z, DCAR
0180; SET CAR CRASHED FLAG
0181 LD A, 01
0182 LD (FLAG), A
0183; DRAW CAR
0184 DCAR CALL DCR
0185; DID CAR CRASH
0186 LD A, (FLAG)
0187 CP 01
0188 JR Z, CRSH
0189; MAKE GAME MORE DIFFICULT
0190; AS SCORE INCREASES
0191; COMPARE SCORE TO MAX SCORE
0192 SPC1 LD DE, (YSCR)
0193 LD HL, (MSCB)
0194 CALL OA39H
0195 CP 01
0196 JR Z, SPC3
0197; INCREASE MAXIMUM SCORE
0198 LD DE, 100
0199 ADD HL, DE
0200 LD (MSCB), HL
0201; COMPARE SPEED TO MIN SPEED
0202 SPC2 LD DE, (SPED)
0203 LD HL, (MINS)
0204 CALL OA39H
0205 CP OBFH
0206 JR NZ, SPC3
0207;
0208;

```

```

0210;
0211; INCREASE SPEED
0212 LD (SPED), HL
0213; INCREASE MINIMUM SPEED
0214 LD DE, 100
0215 SBC HL, DE
0216 LD (MINS), HL
0217; KEEP SPEED DELAY >20H
0218 SPC3 LD HL, (SPED)
0219 LD DE, 20
0220 CALL OA39H
0221 CP 01
0222 JR Z, DLAY
0223 LD (SPED), DE
0224; SPEED CONTROL DELAY
0225 DLAY LD BC, (SPED)
0226 CALL O060H
0227; MAKE NOISES
0228 CALL SOND
0229; CONTINUE GAME
0230 JP SCRL
0231; SIMULATE CAR CRASH
0232; FLASH CARS 50 TIMES
0233 CRSH LD B, 50
0234 IOP5 EXX
0235 LD HL, CAR2
0236 LD DE, (CARL)
0237 LD BC, 2
0238 HALT
0239 LDIR
0240 LD BC, 2000
0241 CALL O060H
0242 CALL DCR
0243 LD BC, 2000
0244 CALL O080H
0245 EXX
0246 DJNZ IOP5
0247 LD BC, 1-2
0248 CALL O060H
0249 XOR A
0250 LD (FLAG), A
0251; WRITE NEW SCORE TO SCREEN
0252 CALL O1C9H
0253 LD HL, MS14
0254 LD DE, VIDE+100
0255 LD BC, 13
0256 LDIR
0257 CALL OA9DH
0258 LD HL, YSCR
0259 CALL O0B1H
0260 CALL OFBDH
0261 LD DE, VIDE+115
0262 LD (7820H), DE
0263 CALL 2B75H
0264; SORT SCORES
0265 CALL SORT
0266; RESET CHAMPION SCORE FLAG
0267 XOR A
0268 LD (MSG), A
0269; WAIT FOR KEYBOARD INPUT
0270 CALL KEYB
0271 JP STRT
0272; GET A RANDOM NUMBER
0273; RETURN INTEGER VALUE IN HL
0274; TO WRA1 = ACC = 7921H
0275 RAND CALL OA9AH
0276; GENERATE A RANDOM NUMBER

```



0417 LD HL,MSG8  
 0418 LD DE,VIDE+228  
 0419 LD BC,22  
 0420 LDIR  
 0428 LD HL,MSG9  
 0428 LD DE,VIDE+260  
 0427 LD BC,22  
 0428 LDIR  
 0429 LD HL,MS10  
 0430 LD DE,VIDE+292  
 0431 LD BC,22  
 0432 LDIR  
 0433 LD HL,MS11  
 0434 LD DE,VIDE+324  
 0435 LD BC,22  
 0436 LDIR  
 0437 LD HL,MS12  
 0438 LD DE,VIDE+356  
 0439 LD BC,22  
 0440 LDIR  
 0441 RET  
 0442;PRINT SCORES  
 0443;PRINT CHAMPION SCORE  
 0444;SET NTF FLAG=2 (INTEGER)  
 0445;CALL OABDH  
 0446 LD HL,CSCR  
 0447;LOAD 4 BYTES FROM THE  
 0448;LOCATION POINTED TO BY HL  
 0449;INTO THE ACCUMULATOR (ACC)  
 0450 CALL O9B1H  
 0451;CONVERTS ACC TO ASCII  
 0452;STRING DELIMITED BY ZERO  
 0453 CALL OFBDH  
 0454 LD DE,VIDE+308  
 0455;CURSOR LOCATION  
 0456 LD (7820H),DE  
 0457;PRINT MESSAGE POINTED TO  
 0458;BY HL. MUST END IN ZERO.  
 0459 CALL 2B75H  
 0460;PRINT CARRIAGE RETURN  
 0461 LD A,ODH  
 0462 CALL O33AH  
 0463;PRINT SECOND SCORE  
 0464 CALL OABDH  
 0465 LD HL,SSCR  
 0466 CALL O9B1H  
 0467 CALL OFBDH  
 0468 LD DE,VIDE+340  
 0469 LD (7820H),DE  
 0470 CALL 2B75H  
 0471;PRINT CARRIAGE RETURN  
 0472 LD A,ODH  
 0473 CALL O33AH  
 0474;PRINT THIRD SCORE  
 0475 CALL OABDH  
 0476 LD HL,TSCR  
 0477 CALL O9B1H  
 0478 CALL OFBDH  
 0479 LD DE,VIDE+372  
 0480 LD (7820H),DE  
 0481 CALL 2B75H  
 0482 RET  
 0483;DRAW VERGE  
 0484;DRAW LD HL,MS13  
 0485 LD DE,VIDE  
 0486 LD BC,32  
 0487;  
 0488;

0491 LDIR  
 0492;DRAW ROAD  
 0493 LD HL,ROAD  
 0494 LD DE,(POSM)  
 0495 LD BC,18  
 0496 LDIR  
 0497 RET  
 0498;COPY LINE 1 TO WHOLE SCREEN  
 0499;COPY LD HL,VIDK  
 0500 LD DE,VIDE+32  
 0501 LD BC,448  
 0502 HALT  
 0503 LDIR  
 0504 RET  
 0505;DRAW CAR  
 0506;DRAW LD HL,CAAR  
 0507 LD DE,(CARL)  
 0508 LD BC,2  
 0509 HALT  
 0510 LDIR  
 0511 RET  
 0512;START OF SCREEN  
 0513;VIDE EQU 7000H  
 0514;CHAMPION FLAG  
 0515;MSG DEFB 0  
 0516;CAR DESIGN  
 0517;CAAR DEFW O979BH  
 0518;OBSTACLE CAR DESIGN  
 0519;CAR1 DEFW O9996U  
 0520;CRASH CAR DESIGN  
 0521;CAR2 DEFW O9D9EH  
 0522;CAR LOCATION ON SCREEN  
 0523;CARL DEFW VIDE+496  
 0524;CHAMPION SCORE  
 0525;CSCR DEFW 2  
 0526 DEFW 0  
 0527;SECOND SCORE  
 0528;SSCR DEFW 0  
 0529 DEFW 0  
 0530;THIRD SCORE  
 0531;TSCR DEFW 0  
 0532 DEFW 0  
 0533;YOUR SCORE  
 0534;YSCR DEFW 0  
 0535 DEFW 0  
 0536;DELAY FOR SPEED CONTROL  
 0537;SPED DEFW 0  
 0538;MAXIMUM SCORE FLAG  
 0539;MSCR DEFW 100  
 0540;MINIMUM SPEED FLAG  
 0541;MINS DEFW 1700H  
 0542;ROAD POSITION ON SCREEN  
 0543;POSN DEFW VIDE+7  
 0544;CAR CRASHED FLAG  
 0545;FLAG DEFB 0  
 0546;ROAD DESIGN  
 0547;ROAD DEFW OBF87H  
 0548 DEFW OBF8FH  
 0549 DEFW OBF8FH  
 0550 DEFW OBF8FH  
 0551 DEFW OBF87H  
 0552 DEFW OBF8FH  
 0553 DEFW OBF8FH  
 0554 DEFW OBF8FH  
 0555;MESSAGES  
 0556;MSG1 EQU \*  
 0557;  
 0558;